Scientific Python Introduction

Iñigo Aldazabal Mensa – Centro de Física de Materiales (CSIC-UPV/EHU)

Software Carpentry Workshop

San Sebastián, 29th March 2016

Materials Physics Center
Centro de Física de Materiales

Universidad del País Vasco
Euskal Herriko Unibertsitatea
CSIC
Consejo Superior de Investigaciones Científicas

Why Python?

```
print("Hello, world!")
```

Why Python?

Python is inherently slow compared to C/C++ or FORTRAN, so why Python for Scientfic Computing?

Python is slow, but...

Syntactically, Python code looks like executable pseudo code. Program development using Python is 5-10 times faster than using C/C++...

The best approach is often to write only the performance-critical parts of the application in C++ or Java, and use Python for all higher-level control and customization.

– Guido van Rossum

Why Python?

We have (for free)

- A general purpose language with a huge spectrum of freely available libraries for almost anything you can think of.
- A very easy to learn (and read) language that smoothly interfaces with C/C++ and FORTRAN (eg. calculation kernels).
- Lots of wrappers for well stablished, fast and long time tested numerical packages.
- Lots of high level utility libraries for scientific computing: plotting, data analisys, parallelization, ...

Why Python? Batteries Included

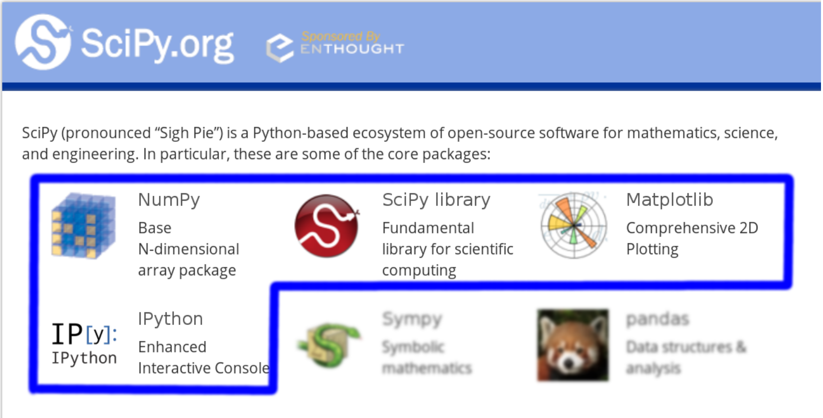
Python Scientific Computing Environment

SciPy.org Sponsored By ENTHOUGHT

SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:

NumPy Base N-dimensional array package	**SciPy library** Fundamental library for scientific computing	**Matplotlib** Comprehensive 2D Plotting
IPython Enhanced Interactive Console	**Sympy** Symbolic mathematics	**pandas** Data structures & analysis

Python Scientific Computing Environment



NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

NumPy

NumPy
Base
N-dimensional
array package

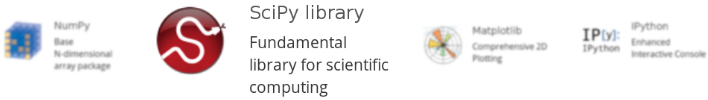
SciPy library
Fundamental
library for scientific
computing

Matplotlib
Comprehensive 2D
Plotting

IPython
Enhanced
Interactive Console

```
import numpy as np

# Create a numpy array, x
X = np.array( [1.1, 1.3, 1.5] )
y = np.sin(a)

# create a random two dimensional numpy array, A
A = np.random.rand(3,3)

A.transpose()
A.trace()
```

SciPy



SciPy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python.

Provides the user with high-level commands and classes for manipulating and visualizing data.

With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling sytems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

SciPy

NumPy
Base
N-dimensional
array package

SciPy library
Fundamental
library for scientific
computing

Matplotlib
Comprehensive 2D
Plotting

IPython
Enhanced
Interactive Console

SciPy subpackages (some of them)

- constants: Physical and mathematical constants
- fftpack: Fast Fourier Transform routines
- integrate: Integration and ordinary differential equation solvers
- interpolate: Interpolation and smoothing splines
- linalg: Linear algebra
- optimize: Optimization and root-finding
- signal: Signal processing
- special: Special functions
- ...

SciPy

SciPy library

Fundamental library for scientific computing

```
import numpy as np
from scipy.special import gamma

x = np.array( [1.1, 2., 3.] )
y = gamma(x)

print (y)
```

```
[ 0.95135077,  1. ,  2. ]
```

matplotlib

NumPy Base N-dimensional array package

SciPy library Fundamental library for scientific computing

Matplotlib Comprehensive 2D Plotting

IPython Enhanced Interactive Console

matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

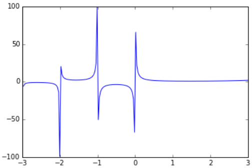
matplotlib can be used in Python scripts, the Python and IPython shell (ala MATLAB® or Mathematica®), web application servers, and several graphical user interface toolkits.

For simple plotting the `pyplot` interface provides a MATLAB-like interface, particularly when combined with the `IPython / Jupyter` environment.

SciPy

Matplotlib
Comprehensive 2D Plotting

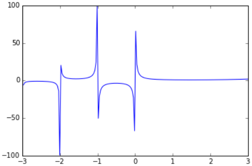
```
import numpy as np
from scipy.special import gamma
import matplotlib.pyplot as plt

x = np.linspace( start=-3., stop
    =3., num=200 )
y = gamma(x)

plt.plot(x,y)
plt.savefig("gamma.png")
```



SciPy

Matplotlib
Comprehensive 2D Plotting

```python
import numpy as np
from scipy.special import gamma
import matplotlib.pyplot as plt

x = np.linspace( start=-3., stop
    =3., num=200 )
y = gamma(x)

plt.plot(x,y)
plt.savefig("gamma.png")
```



Just google matplotlib images for examples!

IPython / Jupyter

IPython
Enhanced
Interactive Console

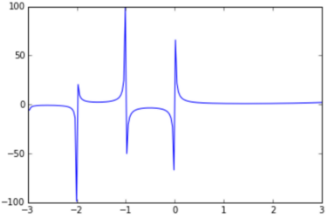
The **IPython / Jupyter Notebook** is a web-based interactive computing system that enables users to author documents that include live code, narrative text, LaTeXequations, HTML, images and video.

These documents contain a full record of a computation and its results and can be shared on email, Dropbox, version control systems (like git/GitHub) or with the Jupyter online notebook viewer nbviewer.jupyter.org.

IPython / Jupyter

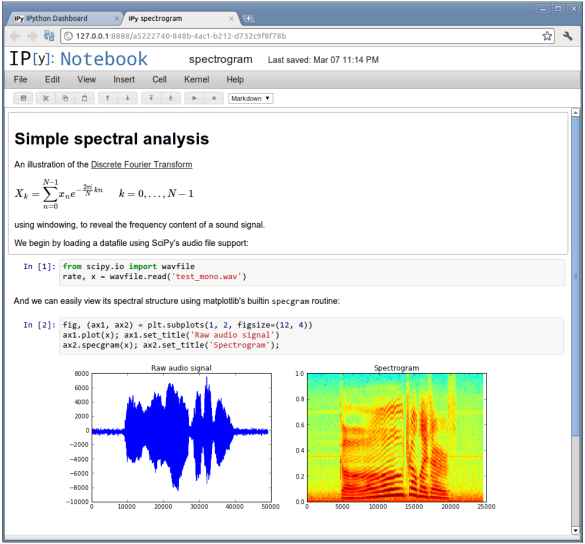
In [45]:
```
%matplotlib inline
import numpy as np
from scipy.special import gamma
import matplotlib.pyplot as plt

x = np.linspace( start=-3., stop=3., num=200 )
y = gamma(x)

plt.plot(x,y)
```

Out[45]: [<matplotlib.lines.Line2D at 0x7fc4ac4660b8>]



In [ ]:

IPython


IPython Dashboard
spectrogram
127.0.0.1:8888/a5222740-848b-4ac1-b212-d732c9f8f78b
IP[y]: Notebook
spectrogram
Last saved: Mar 07 11:14 PM
File
Edit
View
Insert
Cell
Kernel
Help
Markdown
Simple spectral analysis
An illustration of the Discrete Fourier Transform
$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \dots, N-1$
using windowing, to reveal the frequency content of a sound signal.
We begin by loading a datafile using SciPy's audio file support:
In [1]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
And we can easily view its spectral structure using matplotlib's builtin specgram routine:
In [2]: fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
Raw audio signal
Spectrogram

Python Scientific Computing Environment

Let's try it!